OIPE JC46
AUG 1 5 2005
PATENT & TRADEMARK OFFICE

AF
[illegible]

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Tripathi et al.	Attorney Docket No.: SUN1P707/P5232
Application No.: 09/654,103	Examiner: Avi M. Gold
Filed: August 31, 2000	Group: 2157
Title: WEB-SERVER IN-KERNEL INTERFACE TO DATA TRANSPORT SYSTEM AND CACHE MANAGER	Confirmation No.: 5614

CERTIFICATE OF MAILING
I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as first-class mail on August 12, 2005 in an envelope addressed to the Commissioner for Patents, Mail Stop Appeal Brief-Patents, P.O. Box 1450 Alexandria, VA 22313-1450.

Signed: Laura M. Dean
Laura M. Dean

APPEAL BRIEF TRANSMITTAL (37 CFR 192)

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This brief is in furtherance of the Notice of Appeal filed in this case on June 22, 2005.

This application is on behalf of

☐ Small Entity ☒ Large Entity

Pursuant to 37 CFR 1.17(f), the fee for filing the Appeal Brief is:
☐ $250.00 (Small Entity) ☒ $500.00 (Large Entity)

☐ Applicant(s) hereby petition for a _____ extension(s) of time to under 37 CFR 1.136.

If an additional extension of time is required, please consider this a petition therefor.

☐ An extension for months has already been secured and the fee paid therefor of $ is deducted from the total fee due for the total months of extension now requested.

☒ Applicant(s) believe that no (additional) Extension of Time is required; however, if it is determined that such an extension is required, Applicant(s) hereby petition that such an extension be granted and authorize the Commissioner to charge the required fees for an Extension of Time under 37 CFR 1.136 to Deposit Account No. 500388.

Total Fee Due:

Appeal Brief fee	$500.00
Extension Fee (if any)	$
Total Fee Due	$500.00

☒ Enclosed is Check No. 27599 in the amount of $500.00.

☒ Charge any additional fees or credit any overpayment to Deposit Account No. 500388, (Order No. SUN1P707). Two copies of this transmittal are enclosed.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

[illegible]

Alan S. Hodes
Reg. No. 38,185

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300

OIPE JC48
AUG 1 5 2005
PATENT & TRADEMARK OFFICE

CERTIFICATE OF MAILING
I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as first-class mail on August 12, 2005 in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450.

Signed: [illegible]
Laura M. Dean

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Ex Parte Tripathi et al.

Application for Patent: 09/654,103

Filed: August 31, 2000

Application No.:

Group Art Unit 2157

Examiner Avi M. Gold

For:

WEB-SERVER IN-KERNEL INTERFACE TO DATA TRANSPORT SYSTEM AND CACHE MANAGER

APPEAL BRIEF

BEYER WEAVER & THOMAS, LLP
P.O. Box 70250
Oakland, CA 94612-0250
Attorneys for Appellant

08/15/2005 HVUONG1 00000077 09654103
01 FC:1402 500.00 OP

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

[37 CFR 41.37(c)(1)(i)]

The real party in interest is Sun Microsystems, Inc.

2. RELATED APPEALS AND INTERFERENCES

[37 CFR 41.37(c)(1)(ii)]

There are no related appeals or interferences.

3. STATUS OF CLAIMS

[37 CFR 41.37(c)(1)(iii)]

The following claims have been rejected and appealed: claims 1-2, 4, 7, 10, 13-22, 28, 30-31 and 33.

The following claims have been cancelled: claims 3, 5, 8-9, 11-12, 23-27, 29 and 32.

The claims on appeal are reproduced below in Appendix A.

4. STATUS OF AMENDMENTS

[37 CFR 41.37(c)(1)(iv)]

No amendments were filed subsequent to final rejection.

5. SUMMARY OF CLAIMED SUBJECT MATTER

[37 CFR 41.37(c)(1)(v)]

5.1. *Independent Claim 1*

Claim 1 is directed to a method of sending a HTTP request to a HTTP daemon. The method steps take place within a web server.

For example, referring to Fig. 3, a web server 302 is shown. The web server 302 includes a web server/HTTP daemon 310 within the web server 302. As discussed at page 10 of Applicant's specification and as shown in Fig. 3, a network cache accelerator 304 is within the web server 302. Furthermore, the web server 302 also includes a file system 312. In the Fig. 3 example, communication between the network cache accelerator 304 and the clients (client1 100 to clientN 102) is via a Network Interface Card.

In general, the network cache accelerator 304 operates such that, for example, a new stream need not be created for each HTTP request received. See, e.g., page 11, line 11 to page 12, line 24, of Applicant's specification. Fig. 6 is a flowchart that illustrates an example of the method recited in independent claim 1.

Step 602 illustrates "receiving a HTTP request including HTTP request data from a HTTP client." Step 604 illustrates "associating a connection identifier with the HTTP request." Step 606 signifies "repeating the receiving and associating steps for HTTP request from a plurality of HTTP clients." As described at page 13, lines 17-21,

> If it is determined that there are more HTTP requests at block 606, steps 602 and 604 are repeated for one or more HTTP requests. Thus, the NCA continually sends HTTP requests as and when they arrive. In other words, the NCA need not wait for more HTTP requests to be received.

Step 608 illustrates "sending the connection identifier and the associated HTTP request data for the HTTP requests from the HTTP clients in a first stream from a network cache accelerator of the web server to a file system of the web server, the network cache accelerator being adapted for communicating with the HTTP clients." As described at page 13, lines 21-24,

> At block 608, the connection identifier and the associated HTTP request data for the one or more HTTP requests are sent (e.g., by the NCA) in a single stream such as the permanent listing stream described above with reference to Fig. 5.

Step 610 illustrates "storing the HTTP requests with the associated connection identifiers by the file system, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests." As described at page 14, lines 3-6,

> Alternately, the connection identifier and the associated HTTP request data for each of the one or more HTTP requests may then be stored temporarily (e.g., by the NCA file system) at block 610 for retrieval by the HTTP process (i.e.,. the HTTP daemon).

With regard to receiving HTTP responses from the HTTP daemon, Fig. 7, and steps 702 and 703 in particular, for example, illustrating processing in the HTTP daemon to process HTTP requests to obtain HTTP response data (step 702) and to send the HTTP response data to the NCA file system (step 703).

5.2. *Independent Claim 22*

While independent claim 1 basically deals with request data, claim 22 basically deals with response data. Fig. 7 illustrates a method of processing a HTTP response.

Steps 702 and 703 illustrate "receiving HTTP response data from the HTTP daemon." As described at page 14, lines 22-26,

> The web server (i.e., HTTP process) processes the HTTP request to obtain HTTP response data associated with the HTTP request data at block 702. The HTTP process then performs a write by sending HTTP response data to the NCAFS [network cache accelerator file system] at 703.

Step 704 illustrates "obtaining a connection identifier associated with the HTTP response data." As described at page 14, line 26 to page 15, line 1,

> The NCAFS then packages the HTTP response data with the connection identifier identifying one of the HTTP requests received by the web server at block 704.

Step 706 illustrates "creating a stream from a file system of the web server to a network cache accelerator of the web server, the network cache accelerator being adapted for communicating with a plurality of HTTP clients corresponding to a plurality of HTTP requests, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests." As described at page 15, lines 1-3:

> When the HTTP response data has been obtained from the HTTP process, a new stream is created at block 706. For instance, the new stream may be created by the NCA file system.

Step 708 illustrates "sending the HTTP response data and the obtained associated connection identifier corresponding to the plurality of HTTP requests for the plurality of HTTP clients in the stream from the file system of the web server to the network cache accelerator of the web server for transmission to a client" As described at page 15, lines 3-8,

> The HTTP response data and the associated connection identifier are sent in the newly created stream at block 708. For instance, an object may be instantiated in which the HTTP response data and the connection identifier are provided. The NCA (e.g., data transport module) may then intercept and transmit the HTTP response data to the client.

5.3. <u>*Independent Claim 28*</u>

Independent claim 28 is parallel to independent claim 1, but is directed to a computer-readable medium storing computer-readable instructions for performing the steps recited in claim 1. Therefore, in general, the concise explanation of the subject matter of independent claim 1 is a sufficient concise explanation of the subject matter of independent claim 28, and is incorporated herein by reference for that purpose.

5.4. <u>*Independent Claim 30*</u>

Independent claim 30 is similar to independent claim 1, but it is directed to a web server adapted to perform the steps recited in claim 1. Therefore, in general, the concise explanation of the subject matter of independent claim 1 is a sufficient concise explanation of the subject matter of independent claim 30, and is incorporated herein by reference for that purpose.

5.5. <u>*Independent Claim 31*</u>

Independent claim 31 is parallel to independent claim 22, but is directed to a computer-readable medium storing computer-readable instructions for performing the steps recited in claim 22. Therefore, in general, the concise explanation of the subject matter of independent claim 22 is a sufficient concise explanation of the subject matter of independent claim 31, and is incorporated herein by reference for that purpose.

5.6. <u>*Independent Claim 33*</u>

Independent claim 33 is similar to independent claim 22, but it is directed to a web server adapted to perform the steps recited in claim 22. Therefore, in general, the concise explanation of the subject matter of independent claim 22 is a sufficient concise explanation of the subject matter of independent claim 33, and is incorporated herein by reference for that purpose.

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

[37 CFR 41.37(c)(1)(vi)]

Ground 1. Claims 1, 2, 4, 6, 7, 10, 13, 15, 18-26 and 28-33 are rejected under 35 U.S.C. 102(e) as being anticipated by Gupta (US Patent No. 6,374,305).

Ground 2. Claims 14, 16 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gupta in view of Kawabe (US Patent No. 5,968,127).

7. ARGUMENT
[37 CFR 41.37(c)(1)(vii)]

7.1. Ground I

7.1.1. Claim 1

Before discussing the differences between Gupta and the subject matter recited in claim 1, it is useful to briefly review what Gupta discloses. In particular, Gupta discloses (at the portions cited by the Examiner), a mechanism for "packing" communication between a single HTTP client and a web server. For example, see the Abstract, where it is stated that a proxy layer resides on a mobile client station. The proxy layer on the mobile client station captures HTTP requests and packs the HTTP requests for upstream transmission to a server. At the web server, the HTTP request messages are recovered and are then sent to an appropriate web server for further processing.

Similarly, in the web server, responsive HTTP messages are packed for transmission to the mobile client station. The proxy layer on the mobile client station recovers the HTTP responses, which are then sent to the web browser on the mobile client station for processing. As a result, Gupta explains, packed messages are sent in a bandwidth efficient manner over the wireless network. See, for example, col. 4, lines 25-40.

By contrast, the subject matter of Applicant's claim 1 is more closely related to efficient communication *within* a web server, not between a web server and a mobile client.

Applicant does not disagree that Gupta discloses network caching. However, Gupta does not disclose or suggest accelerating network caching. Taking Applicant's claim 1 as an example, network cache acceleration is achieved in a web server by, for multiple HTTP requests (i.e., received from multiple HTTP clients, not a single HTTP client as disclosed by Gupta), sending the multiple HTTP requests in a single stream from a network cache accelerator of the web server to a file system of the web server The multiple HTTP requests recited in claim 1 as being in a single stream are communicated *within* the web server.

By contrast, Gupta does not disclose HTTP requests are received from a plurality of HTTP client applications, wherein the HTTP request data and associated connection identifiers **<u>are sent in a stream from a network cache accelerator of the web server to a file system of the web server</u>**. Put another way, Gupta does not disclose that a "stream" *between a network cache accelerator of the web server and a file system of the web server* corresponds to HTTP requests received from a plurality of HTTP clients. In fact, Gupta does not disclose commingling HTTP requests from different HTTP clients.

The Examiner, in a section of the final rejection entitled "Response to Arguments," states that "Gupta discloses HTTP requests received from a plurality of HTTP client applications. . . " Applicant agrees with this statement (e.g., Fig. 1 of Gupta shows multiple HTTP clients 28 communicating with a server 22), but this statement does not, in fact, address the full extent of Applicant's argument, that Gupta does not disclose the communication, recited in the claim to be <u>within</u> the web server, from a network cache accelerator of the web server to a file system of the web server.

In the final rejection, in apparent "Response" to Applicant's arguments in response to the initial rejection, the Examiner specifically cites to Gupta at col. 6, lines 52-62 and to col. 7, line 66 to column 8, line 10 as alleged support for the anticipation rejection. It is instructive to look at the exact language contained in these cited sections of Gupta:

> **Col. 6, lines 52-62 [which references Gupta's Fig. 3]**
>
> In this regard, information contained in downstream HTTP messages received from a respective web server 44 is also duplicated for storage in the server memory cache 48, or otherwise used to update previously stored information. As information requests are received from any of the client stations 28 in the system 20, the web agent 42 first determines whether the requested information is already present in the server memory cache 48. If so, the information is retrieved from the memory cache 48 by the web agent 42, and transmitted to the respective client station 28, without requiring further processing.
>
> **Col. 7, line 66 to Col. 8, line 10**
>
> For example, while particularly useful for mobile-based client-server systems employing a relatively low speed, high latency wireless network to establish communication links between the server and respective client stations, the advantages of improving efficiency over the client-server communication link and in eliminating the need for specialized APIs as provided by the inventive concepts disclosed and described herein may be obtained in client-server systems using any type of network architecture--e.g., telephonic dial-up or even a high speed LAN connection--wherein raw HTTP message transmission is not possible or is otherwise impractical.

Neither of these portions of Gupta even disclose transmission of any data -- HTTP request data and associated connection identifiers or otherwise -- from a network cache accelerator *of the web server* to a file system *of the web server*. Thus, for at least this reason, the Gupta disclosure is not sufficient to anticipate the subject matter of claim 1.

In addition, claim 1 recites "sending the connection identifier and the associated HTTP request data for the HTTP requests from the HTTP clients **in a first stream**." Gupta discloses receiving information requests from multiple client servers and, when the requested information is already present in the server memory cache, retrieving the requested information from the server memory cache and transmitting the information to the respective client without further processing. Gupta does not disclose sending information in a **stream** let alone, as discussed above, in a stream from a network cache accelerator of the web server to a file system of the web server.

Further on this patent, in the Response filed after final rejection, Applicant made the following statement:

> **If the Examiner continues to reject the claims based on Gupta, then Applicant respectfully requests the Examiner to point out the particular elements of Gupta that are considered to be the <u>network cache accelerator of a web server</u> and the <u>file system of the web server</u>. Applicant further respectfully requests the Examiner to point out the particular disclosure in Gupta of <u>HTTP requests received from a plurality of HTTP client applications, wherein the HTTP request data and associated connection identifiers are sent in a stream</u> from the network cache accelerator of a web server to a file system of the web server.**

In the Advisory Action, the Examiner stated:

> **Regarding the argument to claim 1, the applicant argues that the reference, Gupta, does not disclose sending the connection identifier and the associated HTTP request data for the HTTP request from the HTTP clients in a first stream from a network cache accelerator of the web server to a file system of the web server. The examiner disagrees, as seen in, column 2, lines 39-49, there are packed HTTP messages sent to a web server for processing which inherently includes a file system. The sending of the messages to the server in a stream emulates the same results as the claimed network cache identifier. The packed messages are sent in streams, as shown in column 5, lines 4-11, which would include a first stream. Connection identifiers are attached along with the request data for the HTTP messages/requests as shown in column 6, lines 1-26. It is shown in column 7, lines 66-column 8, lines 10, that multiple clients are used for collecting HTTP requests.**

In the first place, Applicant does not disagree that, perhaps, many web servers include a file system. However, that does not make a file system inherent. In any event, even if the file system can be said to be inherent, it does not immediately follow that the presence of a file system dictates that HTTP requests are sent in a stream from a network cache accelerator of the web server to the inherently-present file system of the web server.

Furthermore, perhaps more significantly, even if it did follow that the presence of a file system dictates that HTTP requests are sent in a stream from a network cache accelerator of the web server to the inherently-present file system of the web server, what would be sent by Gupta in this stream would be requests from a single HTTP client. That is, the requests that are "packed" are requests from a single HTTP client. While Gupta does disclose that multiple HTTP clients can access the web server, only requests from a single HTTP client are packed together. This is what Gupta discloses as making (the wireless communication which, incidentally, is not within the web server, such as between a network cache accelerator of the web server and the file system of the web server) more efficient.

Thus, at best, making all the assumptions made by the Examiner (which Applicant makes only for the purpose of making a good faith attempt to respond to the rejection using the Examiner's characterization of Gupta), it would follow that requests from a single HTTP client are provided along some path within the web server. To the contrary, the recitation of claim 1 makes clear that the HTTP request from the HTTP client and the plurality of other HTTP clients are provided in the stream from the network cache accelerator of the web server to the file system of the web server.

In addition, while the Examiner makes an anticipation rejection, it is noted in the Advisory Action that "The sending of the messages to the server in a stream emulates the same results as the claimed network cache identifier." Even if the Examiner could show this statement were true, "emulation" does not equal anticipation. The Examiner's reliance on "emulation" implies that the Examiner is actually contending that the Gupta disclosure reaches an equivalent result to what is recited in Applicant's claim 1.

An argument of functional equivalence is not a proper one to make in an anticipation rejection. Furthermore, even if the Examiner had made an obviousness rejection, MPEP 2144.06 makes clear that "In order to rely on equivalence as a rationale supporting an obviousness rejection, the equivalency must be recognized in the prior art, and cannot be based on applicant's disclosure or the mere fact that the components at issue are functional or mechanical equivalents." Here, the Examiner has made no showing of equivalency recognized in the prior art. Rather, the Examiner has only baldly asserted that the Gupta reference discloses something that is functionally equivalent (i.e., emulates) to something that is recited in Applicant's claim 1. Thus, for this reason, too, the anticipation rejection of claim 1 is improper.

7.1.2. Claim 2

Claim 2 recites, in addition to the subject matter of claim 1, "creating the first stream." As discussed above with respect to claim 1, Gupta does not even disclose a "stream" from a network cache accelerator of the web server to a file system of the web server. Certainly, Gupta does not disclose "creating" the stream.

7.1.3. Claim 4

Claim 4 recites "creating a second stream" and "sending the HTTP response data and the connection identifier in the second stream from the file system to the network cache accelerator." The Examiner contends that col. 2, lines 39-49 of Gupta discloses this feature. This portion of Gupta discloses:

> **From the server message handler, the web agent captures the packed messages and recovers the original, (i.e., "raw") HTTP messages. The HTTP messages are then transmitted by the web agent to an appropriate web server for processing. Responsive HTTP messages (e.g., containing HTML, binary, java, or other types of information) transmitted from a respective web server are captured by the web agent, which packs the messages into the selected communication transmission format. The packed messages are then forwarded from the web agent to the server message handler for transmission to the client station.**

Nothing in this portion of Gupta discloses sending HTTP response data and the connection identifier in a stream from the file system to the network cache accelerator. While this portion does disclose "The packed messages are then forwarded from the web agent to the server message handler for transmission the station," this does not meet the feature recited in claim 4.

7.1.4. Claim 6

In the first place, Gupta does not disclose creating a second stream, as discussed above. Furthermore, there is no disclosure that creating the second stream , for handling response data (it is not even clear what portion of Gupta is considered to be the "second stream") is performed in parallel with reading of a HTTP request and preparation of a corresponding HTTP request by the HTTP daemon.

7.1.5. *Claim 7*

As discussed above relative to claim 6, Gupta does not disclose creating a second stream. Further, there is no disclosure that creating the second stream is performed asynchronously with the reading of the HTTP request and the preparation of the corresponding HTTP request by the HTP daemon.

7.1.6. *Claim 10*

The Examiner again relies on Gupta's disclosure at col. 2, lines 39-49, as allegedly disclosing the features of this claim – i.e., instantiating an object, providing the connection identifier and the associated HTTP request data for the HTTP requests in the object, and sending the connection identifier and the associated HTTP request data comprises sending the object.

This cited portion of Gupta deals with creating packed messages and forwarding the packed messages from the web agent to the server message handler for transmission to the client station. There is no disclosure at all of instantiating an object or otherwise dealing with an object.

7.1.7. *Claim 13*

The Examiner again relies on Gupta's disclosure at col. 2, lines 39-49, as allegedly disclosing the features of this claim. While the cited portion discloses that a read request is provided from "the web agent to an appropriate web server for processing," this does not disclose "sending HTTP request data from the file system to the HTTP daemon in response to the read request (i.e., from the HTTP daemon). The cited portion of Gupta says nothing about a HTTP daemon (perhaps the "web server" as alleged by the Examiner?) making a "read request," to a file system of the web server or otherwise.

7.1.8. *Claim 15*

For this feature, the Examiner again relies on Gupta's disclosure at col. 2, lines 39-49. There is nothing in Gupta that discloses a file system, let alone receiving at a file system, from an HTTP daemon, HTTP response data associated with an HTTP request.

7.1.9. *<u>Claim 18</u>*

This claim recites storing the HTTP response data. The Examiner points to a portion of Gupta that discloses storing "information received from the server" in the client station's memory cache. The feature of claim 18 occurs in the web server. Thus, the cited portion of Gupta fails to disclose the feature of claim 18.

7.1.10. *<u>Claim 19</u>*

This claim recites "sending a write command including the connection identifier and the HTTP response data to a data transport module capable of transmitting the HTTP response data to a client." The Examiner again relies on the same eleven-line portion of Gupta.

Nothing in this portion of Gupta discloses a "write command," the content of a write command as "including the connection identifier and the HTTP response data," let alone a "data transport module" that receives the write command.

7.1.11. *<u>Claim 20</u>*

This claim is similar to claim 4 (except for its dependencies). Applicant's arguments set forth above with respect to claim 4 are incorporated herein by reference.

7.1.12. *<u>Claim 21</u>*

This claim is similar to claim 10 (except for its dependencies). Applicant's arguments set forth above with respect to claim 10 are incorporated herein by reference.

7.1.13. *<u>Claim 22</u>*

Claim 22 is similar to claim 1, except that is relates to "response data" as opposed to "request data." Similar to that discussed above with respect to claim 1, Gupta does not disclose HTTP response data "for a plurality of HTTP client corresponding to a plurality of HTTP requests" is sent in a stream from the file system of a web server to a network cache accelerator of the web server." Like with the request data, Gupta does not disclose a "stream" between a file system of a web server and a network cache accelerator of the web server, particularly a stream in which responses destined for a plurality of client are intermingled together.

7.1.14. Claim 30

Contrary to what the Examiner appears to be asserting, claim 30 is more analogous to claim 1 (with respect to request data) than to claim 22 (with respect to response data). Applicant's remarks with respect to claim 1 are incorporated herein by reference in refutation of the rejection of claim 30.

7.1.15. Claim 31

Claim 31 is analogous to claim 22. Applicant's remarks above with respect to claim 22 are incorporated herein by reference.

7.1.16. Claim 33

Claim 33 is analogous to claim 22 as well, and Applicant's remarks above with respect to claim 22 are incorporated herein by reference.

7.2. Ground II

7.2.1. Claims 14, 16 and 17

For the feature specifically recited in claims 14, 16 and 17, the Examiner relies on a secondary reference – the Kawabe patent. The Examiner contends that Kawabe discloses the claim feature. The Examiner's stated motivation for modifying Gupta in view of Kawabe is that "it would result in an efficient method to send and receive data."

The motivation relied upon in making an obviousness rejection must come from knowledge held by one of ordinary skill in the art or from the references themselves. To the extent the Examiner may be correct that the combination yields what is claimed (it is contended, of course, that the Examiner's foundational allegations with respect to Gupta are incorrect, as set forth in great detail above), then the Examiner is using impermissible hindsight in proposing the modification to Gupta in view of Kawabe. The Examiner has not make any showing whatsoever that one of ordinary skill in the art would be motivated to make the modification, beyond the bald assertion of "an efficient method." It is submitted that the rejection of claims xxx under 35 U.S.C. §103 is improper and should be withdrawn. Accordingly, it is respectfully requested that the Board reverse the Examiner's rejection and remand the application to the Examiner with directions to allow all claims.

8. CONCLUSION

In view of the foregoing, it is respectfully submitted that the Examiner's rejection of claims 1, 2, 4, 6, 7, 10, 13, 15, 18-26 and 25-28 as being anticipated by Gupta, and rejection of claims 14, 16, and 17 as being unpatentable over Gupta, is erroneous. Accordingly, the rejection of claims 1, 2, 4, 6, 7, 10, 13, 15, 18-26 and 25-28 under 35 U.S.C. 102(e) and claims 14, 16, and 17 under 35 U.S.C. §103(a) should be reversed.

Respectfully submitted,

[illegible]

Alan S. Hodes
Registration No. 38,185

BEYER, WEAVER & THOMAS LLP
Attorneys for Appellant

9. CLAIMS APPENDIX
[37 CFR 41.37(c)(1)(viii)]

CLAIMS ON APPEAL

1. (Previously Presented) In a web server, a method of sending a HTTP request to a HTTP daemon, comprising:

receiving a HTTP request including HTTP request data from a HTTP client;

associating a connection identifier with the HTTP request;

repeating the receiving and associating steps for HTTP requests from a plurality of other HTTP clients;

sending the connection identifier and the associated HTTP request data for the HTTP requests from the HTTP clients in a first stream from a network cache accelerator of the web server to a file system of the web server, the network cache accelerator being adapted for communicating with the HTTP clients; and

storing the HTTP requests with the associated connection identifiers by the file system, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests.

2. (Previously Presented) The method as recited in claim 1, further comprising:

creating the first stream;

wherein sending the connection identifier and the associated HTTP request data for the one or more HTTP requests from the HTTP clients comprises sending the connection identifier and the associated HTTP request data for the one or more HTTP requests in the first stream.

3. (Cancelled)

4. (Previously Presented) The method as recited in claim 2, further comprising:

creating a second stream from the file system of the web server to the network cache accelerator of the web server;

obtaining HTTP response data associated with one of the HTTP requests by the file system from the HTTP daemon; and

sending the HTTP response data and the connection identifier in the second stream from the file system to the network cache accelerator.

5. (Cancelled)

6. (Previously Presented) The method as recited in claim 4, wherein creating the second stream is performed in parallel with reading of an HTTP request and preparation of a corresponding HTTP response by the HTTP daemon.

7. (Previously Presented) The method as recited in claim 4, wherein creating the second stream is further performed asynchronously with the reading of the HTTP request and the preparation of the corresponding HTTP response by the HTTP daemon.

8. (Cancelled)

9. (Cancelled)

10. (Previously Presented) The method as recited in claim 1, further comprising:

instantiating an object;

providing the connection identifier and the associated HTTP request data for the HTTP requests in the object; and

wherein sending the connection identifier and the associated HTTP request data for the one or more HTTP requests comprises sending the object.

11. (Cancelled)

12. (Cancelled)

13. (Previously Presented) The method as recited in claim 1, further comprising:

receiving a read request at the file system from the HTTP daemon;

sending HTTP request data from the file system to the HTTP daemon in response to the read request.

14. (Previously Presented) The method as recited in claim 13, wherein sending HTTP request data from the file system to the HTTP daemon in response to the read request comprises:

sending a file descriptor including the HTTP request data, the file descriptor having a private attachment including the connection identifier associated with the HTTP request data.

15. (Previously Presented) The method as recited in claim 13, further comprising:

receiving HTTP response data associated with the HTTP request data at the file system from the HTTP daemon.

16. (Previously Presented) The method as recited in claim 15, wherein receiving HTTP response data associated with the HTTP request data at the file system from the HTTP daemon comprises:

receiving a file descriptor including the HTTP response data, the file descriptor having a private attachment including the connection identifier associated with the HTTP request data.

17. (Original) The method as recited in claim 16, further comprising:

obtaining the connection identifier from the private attachment; and

storing the HTTP response data such that the HTTP response data is associated with one of the HTTP requests and the obtained connection identifier.

18. (Original) The method as recited in claim 15, further comprising:

storing the HTTP response data such that the HTTP response data is associated with one of the HTTP requests and the associated connection identifier.

19. (Original) The method as recited in claim 15, further comprising:

sending a write command including the connection identifier and the HTTP response data to a data transport module capable of transmitting the HTTP response data to a client.

20. (Previously Presented) The method as recited in claim 15, further comprising:

creating a second stream from the file system to the network cache accelerator; and

sending the HTTP response data and the connection identifier in the second stream from the file system to the network cache accelerator.

21. (Previously Presented) The method as recited in claim 20, further comprising:

instantiating an object;

providing the HTTP response data and the connection identifier in the object; and

wherein sending the HTTP response data and the connection identifier comprises sending the object to a data transport module of the network cache accelerator for transmission to a client.

22. (Previously Presented) In a web server, a method of processing a HTTP response including HTTP response data received from a HTTP daemon, comprising:

receiving HTTP response data from the HTTP daemon;

obtaining a connection identifier associated with the HTTP response data;

creating a stream from a file system of the web server to a network cache accelerator of the web server, the network cache accelerator being adapted for communicating with a plurality of HTTP clients corresponding to a plurality of HTTP requests, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests; and

sending the HTTP response data and the obtained associated connection identifier corresponding to the plurality of HTTP requests for the plurality of HTTP clients in the stream from the file system of the web server to the network cache accelerator of the web server for transmission to a client.

23. - 27. (Cancelled)

28. (Previously Presented) A computer-readable medium storing thereon computer-readable instructions for sending a plurality of HTTP requests to a HTTP daemon in a web server, comprising:

instructions for receiving a HTTP request including HTTP request data from a HTTP client;

instructions for associating a connection identifier with the HTTP request;

instructions for repeating the receiving and associating steps for HTTP requests from a plurality of other HTTP clients;

instructions for sending the connection identifier and the associated HTTP request data for the HTTP requests from the HTTP clients in a first stream from a network cache accelerator of the web server to a file system of the web server, the network cache accelerator being adapted for communicating with the HTTP clients; and

instructions for storing the HTTP requests with the associated connection identifiers by the file system, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests.

29. (Cancelled)

30. (Previously Presented) A web server adapted for sending a HTTP request to a HTTP daemon, comprising:

a processor; and

a memory, at least one of the processor and the memory being adapted for:

receiving a HTTP request including HTTP request data from a HTTP client;

associating a connection identifier with the HTTP request;

repeating the receiving and associating steps for HTTP requests from a plurality of other HTTP clients;

sending the connection identifier and the associated HTTP request data for the HTTP requests from the HTTP clients in a first stream from a network cache accelerator of the web server to a file system of the web server, the network cache accelerator being adapted for communicating with the HTTP clients; and storing the HTTP requests with the associated connection identifiers by the file system, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests.

31. (Previously Presented) A computer-readable medium storing thereon computer-readable instructions for processing a HTTP response including HTTP response data received from a HTTP daemon in a web server, comprising:

instructions for receiving HTTP response data from the HTTP daemon;

instructions for obtaining a connection identifier associated with the HTTP response data;

instructions for creating a stream from a file system of the web server to a network cache accelerator of the web server, the network cache accelerator being adapted for communicating with a plurality of HTTP clients corresponding to a plurality of HTTP requests, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests; and

instructions for sending the HTTP response data and the obtained associated connection identifier corresponding to the plurality of HTTP requests in the stream from the file system of the web server to the network cache accelerator of the web server for transmission to a client.

32. (Cancelled)

33. (Previously Presented) A web server adapted for processing a HTTP response including HTTP response data received from a HTTP daemon, comprising:

a processor; and

a memory, at least one of the processor and the memory being adapted for:

receiving HTTP response data from the HTTP daemon;

obtaining a connection identifier associated with the HTTP response data;

creating a stream from a file system of the web server to a network cache accelerator of the web server, the network cache accelerator being adapted for communicating with a plurality of HTTP clients corresponding to a plurality of

HTTP requests, the file system being adapted for sending each of the HTTP requests to the HTTP daemon and receiving HTTP responses from the HTTP daemon for each of the HTTP requests; and

sending the HTTP response data and the obtained associated connection identifier corresponding to the plurality of HTTP requests in the stream from the file system of the web server to the network cache accelerator of the web server for transmission to a client.